# Story

This assignment is a list of exercises for practicing creation of Web applications. You can find multiple features which you can accomplish separately after the Preparation part. Do not forget to commit after you completed a part.

# What you are going to learn

* How to create Web API
* How to register services in asp.net core applications
* How to use extension method to simplify registration of services
* How to modify default routing of API controllers
* How to test your API with postman
* Get familiar with one of multithreading problems – race condition and know basic way of prevention

# Tasks

## Create ASP.NET Core API application

Create ASP.NET Core API application project inside solution. Have a look at separate RequestCounter.Services csproj inside solution for holding logic – add project reference in your api project to this project.

## Create main endpoint

* Create RequestCounterController which provides http methods GET, POST, PUT, PATCH, DELETE on each return status code 200 with no data in it.
* Use postman to test each http method execution

## Create route to request counter

* Modify routing to expose request counter endpoint under following url [base url]/api/request-counter, test using Postman new routing
* Using dependency inversion pattern add IRequestCountStatsService to RequestCounter controller
* Use appropriate methods of injected interface in endpoint methods
* Inside RequestCountStatisticsService implement IRequestCountStatsService method IncreaseCounter, leave GetStatistics not implemented (it should throw NotImplemented exception when one tries to execute method) – store statistics in-memory for now.
* Ensure you protect IncreaseCounter method from counting other methods names than allowed ones
* Register service dependency as singleton
* Use IncreaseCounter method in Request Counter controller methods
* Use postman to test and debug to check if counter increases properly and its state is not reset between requests.
* Answer question – what is the difference between AddScoped, AddTransient, AddSingleton ?

## Create statistics endpoint

* Create new Statistics controller
* Created endpoint available under [url]/api/statistics
* Expose method GET which will return statistics
* Inject IRequestCountStatsService into newly created controller
* Use inside method of controller appropriate method of the interface
* Implement method inside class implementing IRequestCountStatsService
* Implement extension method inside class AppServiceExtension which allows register dependency outside of Startup.cs file – once done use it to register service

## Save counts

* Modify your implementation of IRequestCountStatsService to store and read counted requests in file request-count.txt through DataReader class (see properties of the file in solution explorer – what CopyAlways does? What Copy if newer does ?)
* Test you solution by running it, counting some requests, stop application and run it again
* Ensure your code is thread-safe – address situation when multiple requests are executed at same time – how to prevent race condition

# Links

<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/extension-methods>

<https://docs.microsoft.com/en-us/aspnet/core/web-api/?view=aspnetcore-5.0>

<https://docs.microsoft.com/en-us/aspnet/core/fundamentals/dependency-injection?view=aspnetcore-5.0>

<https://www.postman.com/downloads/>

<https://www.guru99.com/postman-tutorial.html>

<https://en.wikipedia.org/wiki/Race_condition>

<https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/lock-statement>